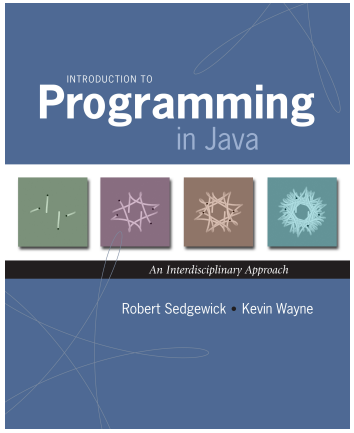
1.5 Input and Output



Input and Output

Input devices.


Keyboard


Mouse


Hard drive


Network


Digital camera


Microphone

Output devices.


Display


Speakers


Hard drive


Network


Printer


MP3 Player

Goal. Java programs that interact with the outside world.

Input and Output

Input devices.


Keyboard


Mouse


Hard drive


Network


Digital camera


Microphone

Output devices.


Display


Speakers


Hard drive


Network


Printer

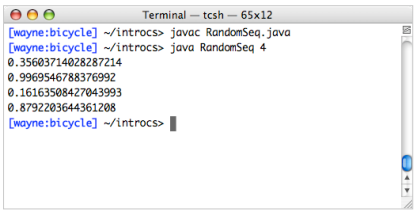

MP3 Player

Our approach.

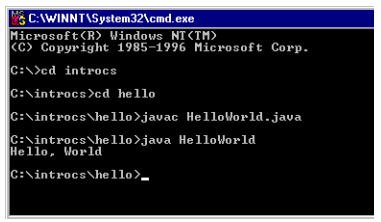
- Define Java libraries of functions for input and output.
- Use operating system (OS) to connect Java programs to:
  file system, each other, keyboard, mouse, display, speakers.

Terminal

Terminal. Application where you can type commands to control the operating system.



Mac OS X



Microsoft Windows

Command-Line Input and Standard Output

Command-line input. Read an integer `N` as command-line argument.

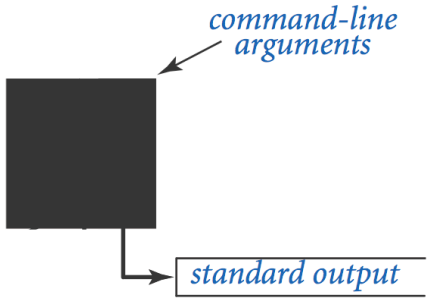
Standard output.

- Flexible OS abstraction for output.
- In Java, output from `System.out.println()` goes to standard output.
- By default, standard output is sent to Terminal.

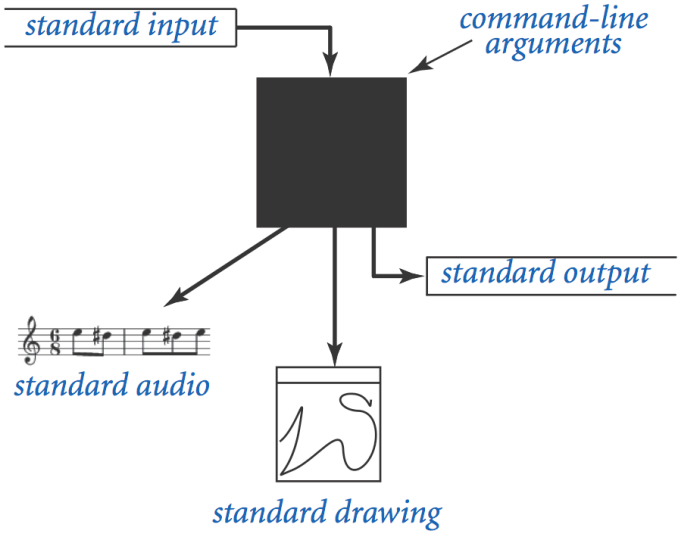
```
public class RandomSeq {
   public static void main(String[] args) {
      int N = Integer.parseInt(args[0]);
      for (int i = 0; i < N; i++) {
         System.out.println(Math.random());
      }
   }
}
```

```
% java RandomSeq 4
0.9320744627218469
0.4279508713950715
0.08994615071160994
0.6579792663546435
```


Old Bird's Eye View



New Bird's Eye View



Standard Input and Output

Command-Line Input vs. Standard Input

Command-line inputs.

- Use command-line inputs to read in a few user values.
- Not practical for many user inputs.
- Input entered before program begins execution.

Standard input.

- Flexible OS abstraction for input.
- By default, standard input is received from Terminal window.
- Input entered while program is executing.

Standard Input and Output

Standard input. `StdIn` is library for reading text input.
Standard output. `StdOut` is library for writing text output.

`public class StdIn`

`boolean isEmpty()`	*`true` if no more values, `false` otherwise*
`int readInt()`	*read a value of type `int`*
`double readDouble()`	*read a value of type `double`*
`long readLong()`	*read a value of type `long`*
`boolean readBoolean()`	*read a value of type `boolean`*
`char readChar()`	*read a value of type `char`*
`String readString()`	*read a value of type `String`*
`String readLine()`	*read the rest of the line*
`String readAll()`	*read the rest of the text*

`public class StdOut`

`void print(String s)`	*print `s`*
`void println(String s)`	*print `s`, followed by newline*
`void println()`	*print a new line*
`void printf(String f, ... )`	*formatted print*

libraries developed for this course (also broadly useful)

Standard Input and Output

To use. Download `StdIn.java` and `StdOut.java` from booksite, and put in working directory (or use classpath).

see booksite

```
public class Add {
   public static void main(String[] args) {
      StdOut.print("Type the first integer: ");
      int x = StdIn.readInt();
      StdOut.print("Type the second integer: ");
      int y = StdIn.readInt();
      int sum = x + y;
      StdOut.println("Their sum is " + sum);
   }
}
```

```
% java Add
Type the first integer: 1
Type the second integer: 2
Their sum is 3
```


Averaging A Stream of Numbers

Average. Read in a stream of numbers, and print their average.

```
public class Average {
   public static void main(String[] args) {
      double sum = 0.0;  // cumulative total
      int n = 0;         // number of values
      while (!StdIn.isEmpty()) {
         double x = StdIn.readDouble();
         sum = sum + x;
         n++;
      }
      StdOut.println(sum / n);
   }
}
```

```
% java Average
10.0 5.0  6.0
 3.0 7.0 32.0
<Ctrl-d>
10.5
```

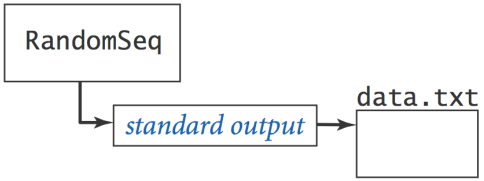
<Ctrl-d> for OS X/Linux/Unix/DrJava
<Ctrl-z> for Windows

Key point. Program does not limit the amount of data.

Redirection and Piping

Redirecting Standard Output

Redirecting standard output. Use OS directive to send standard output to a file for permanent storage (instead of terminal window).

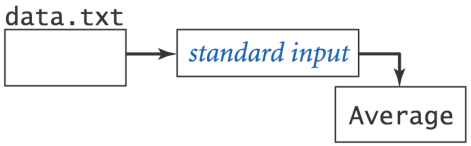



```
% java RandomSeq 1000 > data.txt
```

redirect stdout

Redirecting Standard Input

Redirecting standard input. Use OS directive to read standard input from a file (instead of terminal window).

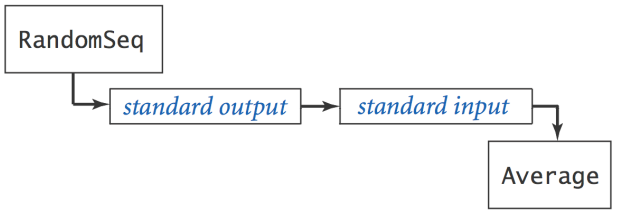



```
% more < data.txt
0.5475375782884312
0.4971087292684019
0.23123808041753813
...

% java Average < data.txt
0.4947655567740991
```

redirect stdin

Connecting Programs

Piping. Use OS directive to make the standard output of one program become the standard input of another.



pipe stdout of `RandomSeq` to stdin of `Average`

```
% java RandomSeq 1000000 | java Average
0.4997970473016028

% java RandomSeq 1000000 | java Average
0.5002071875644842
```

Standard Drawing

Standard drawing. `StdDraw` is library for producing graphical output.

```
public class StdDraw
```

```
void line(double x0, double y0, double x1, double y1)
void point(double x, double y)
void text(double x, double y, String s)
void circle(double x, double y, double r)
void filledCircle(double x, double y, double r)
void square(double x, double y, double r)
void filledSquare(double x, double y, double r)
void polygon(double[] x, double[] y)
void filledPolygon(double[] x, double[] y)
```

`void setXscale(double x0, double x1)`	*reset x range to* (x_0, x_1)
`void setYscale(double y0, double y1)`	*reset y range to* (y_0, y_1)
`void setPenRadius(double r)`	*set pen radius to* `r`
`void setPenColor(Color c)`	*set pen color to* `c`
`void setFont(Font f)`	*set text font to* `f`
`void setCanvasSize(int w, int h)`	*set canvas to* `w`*-by-*`h` *window*
`void clear(Color c)`	*clear the canvas; color it* `c`
`void show(int dt)`	*show all; pause* `dt` *milliseconds*
`void save(String filename)`	*save to a* `.jpg` *or* `w.png` *file*

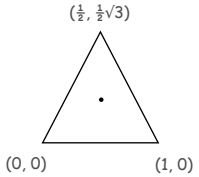
Note: Methods with the same names but no arguments reset to default values.

library developed for this course (also broadly useful)

Standard drawing. We provide library `stdDraw` to plot graphics.
To use. Download `stdDraw.java` and put in working directory.

```
public class Triangle {
   public static void main(String[] args) {
      double t = Math.sqrt(3.0) / 2.0;
      StdDraw.line(0.0, 0.0, 1.0, 0.0);
      StdDraw.line(1.0, 0.0, 0.5,   t);
      StdDraw.line(0.5,   t, 0.0, 0.0);
      StdDraw.point(0.5, t/3.0);
   }
}
```

```
% java Triangle
```



Plot filter. Read in a sequence of (x, y) coordinates from standard input, and plot using standard drawing.

```
public class PlotFilter {
   public static void main(String[] args) {
      double xmin = StdIn.readDouble();
      double ymin = StdIn.readDouble();
      double xmax = StdIn.readDouble();
      double ymax = StdIn.readDouble();
      StdDraw.setXscale(xmin, xmax);
      StdDraw.setYscale(ymin, ymax);

      while (!StdIn.isEmpty()) {
         double x = StdIn.readDouble();
         double y = StdIn.readDouble();
         StdDraw.point(x, y);
      }
   }
}
```

← rescale coordinate system

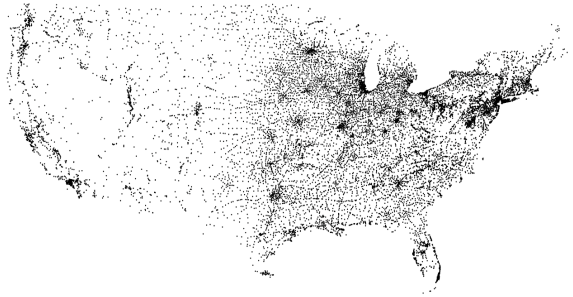
← read in points, and plot them

Data Visualization

```
% more < USA.txt
669905.0 247205.0 1244962.0 490000.0
 1097038.8890  245552.7780
 1103961.1110  247133.3330
 1104677.7780  247205.5560
 ...

% java PlotFilter < USA.txt
```

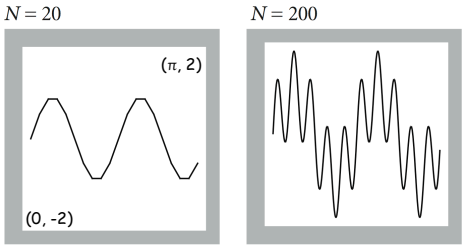
bounding box

coordinates of 13,509 US cities



Plotting a Function

```
double[] x = new double[N+1];
double[] y = new double[N+1];
for (int i = 0; i <= N; i++) {
   x[i] = Math.PI * i / N;
   y[i] = Math.sin(4*x[i]) + Math.sin(20*x[i]);
}
StdDraw.setXscale(0, Math.PI);
StdDraw.setYscale(-2.0, +2.0);
for (int i = 0; i < N; i++)
   StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
```



$$y = \sin 4x + \sin 20x, \quad x \in [0, \pi]$$

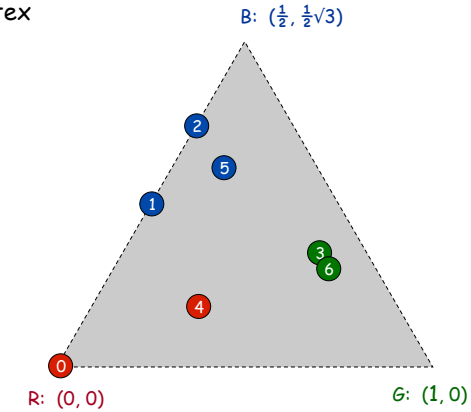
Chaos Game

Chaos game. Play on equilateral triangle, with vertices R, G, B.

- Start at R.
- Repeat the following N times:
  - pick a random vertex
  - move halfway between current point and vertex
  - draw a point in color of vertex

Q. What picture emerges?

B B G R B G ...



Chaos Game

```
public class Chaos {
   public static void main(String[] args) {
      int T = Integer.parseInt(args[0]);
      double[] cx = { 0.000, 1.000, 0.500 };
      double[] cy = { 0.000, 0.000, 0.866 };

      double x = 0.0, y = 0.0;
      for (int t = 0; t < T; t++) {
         int r = (int) (Math.random() * 3);
         x = (x + cx[r]) / 2.0;
         y = (y + cy[r]) / 2.0;
         StdDraw.point(x, y);
      }
   }
}
```

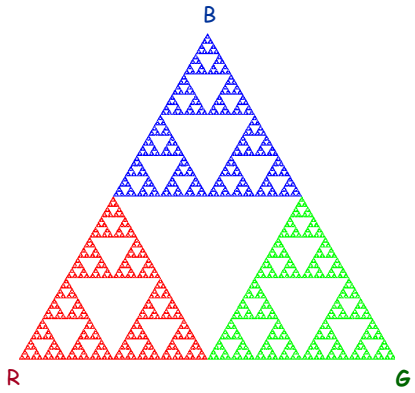
$\frac{1}{2}\sqrt{3}$ (avoid hardwired constants like this)

between 0 and 2

Chaos Game

Easy modification. Color point according to random vertex chosen using `StdDraw.setPenColor(StdDraw.RED)` to change the pen color.

```
% java Chaos 10000
```

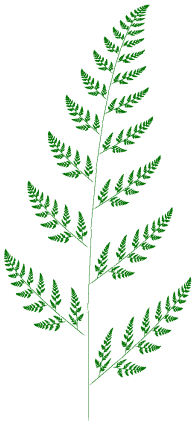


Sierpinski triangle

Barnsley Fern

Barnsley fern. Play chaos game with different rules.

probability	new x	new y
2%	.50	.27y
15%	-.14x + .26y + .57	.25x + .22y - .04
13%	.17x - .21y + .41	.22x + .18y + .09
70%	.78x + .03y + .11	-.03x + .74y + .27



Q. What does computation tell us about nature?
Q. What does nature tell us about computation?

20th century sciences. Formulas.
21st century sciences. Algorithms?

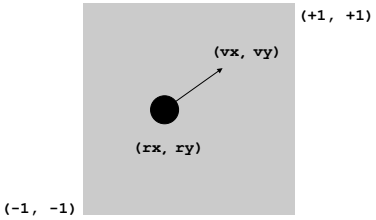
Animation

Animation loop. Repeat the following:

- Clear the screen.
- Move the object.
- Draw the object.
- Display and pause for a short while.

Ex. Bouncing ball.

- Ball has position `(rx, ry)` and constant velocity `(vx, vy)`.
- Detect collision with wall and reverse velocity.



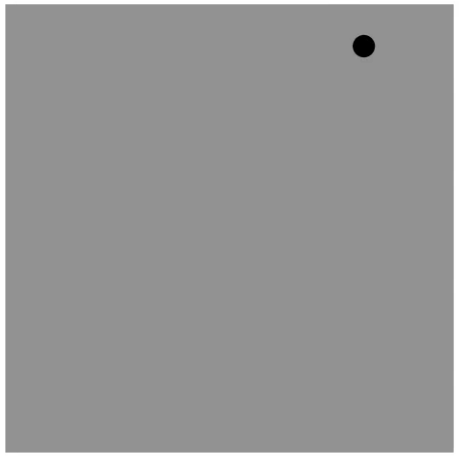
Bouncing Ball

```
public class BouncingBall {
   public static void main(String[] args) {
      double rx = .480, ry = .860;                  position
      double vx = .015, vy = .023;                  constant velocity
      double radius = .05;                          radius

      StdDraw.setXscale(-1.0, +1.0);                rescale coordinates
      StdDraw.setYscale(-1.0, +1.0);

      while(true) {
         if (Math.abs(rx + vx) + radius > 1.0) vx = -vx;   bounce
         if (Math.abs(ry + vy) + radius > 1.0) vy = -vy;

         rx = rx + vx;
         ry = ry + vy;                              update position

         StdDraw.setPenColor(StdDraw.GRAY);         clear background
         StdDraw.filledSquare(0.0, 0.0. 1.0);
         StdDraw.setPenColor(StdDraw.BLACK);        draw the ball
         StdDraw.filledCircle(rx, ry, radius);
         StdDraw.show(20);
      }
   }
}
```

turn on animation mode:
display and pause for 50ms

Bouncing Ball Demo

```
% java BouncingBall
```



Special Effects

Images. Put `.gif`, `.png`, or `.jpg` file in the working directory and use `StdDraw.picture()` to draw it.

Sound effects. Put `.wav`, `.mid`, or `.au` file in the working directory and use `StdAudio.play()` to play it.

Ex. Modify `BouncingBall` to display image and play sound upon collision.

- Replace `StdDraw.filledCircle()` with:

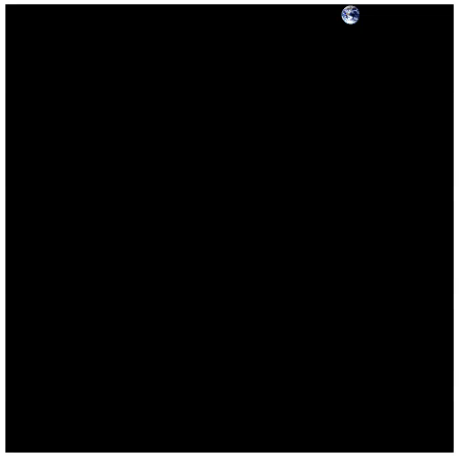
```
StdDraw.picture(rx, ry, "earth.gif");
```

- Add following code upon collision with wall:

```
StdAudio.play("boing.wav");
```


Deluxe Bouncing Ball Demo

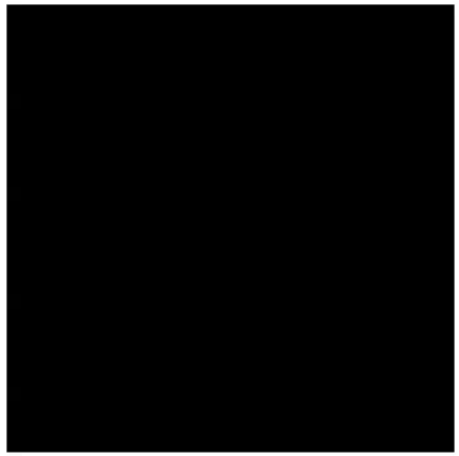
```
% java DeluxeBouncingBall
```



Bouncing Ball Challenge

Q. What happens if you call `StdDraw.filledSquare()` once before loop (instead of inside)?

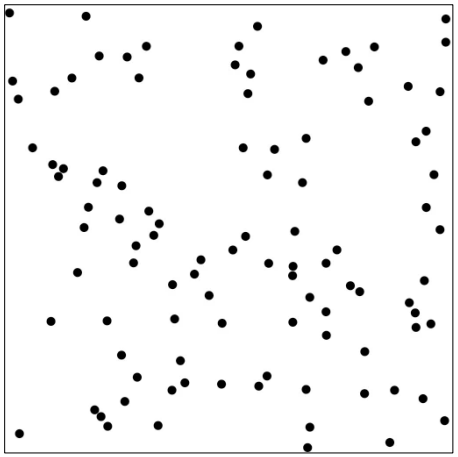
```
% java DeluxeBouncingBall
```



Colliding Balls

Challenge. Add elastic collisions.

```
% java CollidingBalls 100
```



N-body Simulation

Challenge. Add gravity.

```
% java NBody < planets.txt
```